AWS Whitepaper

AWS Fault Isolation Boundaries

AWS Fault Isolation Boundaries: AWS Whitepaper

Table of Contents

Abstract and introduction 1
Abstract 1
Are you Well-Architected? 1
Introduction 1
AWS infrastructure 3
Availability Zones 3
Regions 4
AWS Local Zones 5
AWS Outposts 5
Points of presence 6
Partitions 7
Control planes and data planes 7
Static stability 8
Summary 9
AWS service types 10
Zonal services 10
Regional services 13
Global services 14
Global services that are unique by partition 14
Global services in the edge network 16
Global Single-Region operations 17
Services that use default global endpoints 21
Global services summary 23
Conclusion 26
Appendix A - Partitional service guidance 27
AWS IAM 27
AWS Organizations 27
AWS Account Management 28
Route 53 Application Recovery Controller 29
AWS Network Manager 29
Route 53 Private DNS 30
Appendix B - Edge network global service guidance 31
Route 53 31
Amazon CloudFront 31

Amazon Certificate Manager 32
AWS Web Application Firewall (WAF) and WAF Classic 32
AWS Global Accelerator 32
Amazon Shield Advanced 33
Appendix C - Single-Region services 34
Contributors 35
Document revisions 36
AWS Glossary 37
Notices 38

AWS Fault Isolation Boundaries

Publication date: **November 16, 2022** (*Document revisions*)

Abstract

Amazon Web Services (AWS) provides different isolation boundaries, such as Availability Zones (AZ), Regions, control planes, and data planes. This paper details how AWS uses these boundaries to create zonal, Regional, and global services. It also includes prescriptive guidance on how to consider dependencies on these different services and how to improve the resilience of workloads you build using them.

Are you Well-Architected?

The AWS Well-Architected Framework helps you understand the pros and cons of the decisions you make when building systems in the cloud. The six pillars of the Framework allow you to learn architectural best practices for designing and operating reliable, secure, efficient, cost-effective, and sustainable systems. Using the AWS Well-Architected Tool, available at no charge in the AWS Management Console, you can review your workloads against these best practices by answering a set of questions for each pillar.

For more expert guidance and best practices for your cloud architecture—reference architecture deployments, diagrams, and whitepapers—refer to the AWS Architecture Center.

Introduction

AWS operates a global infrastructure to provide cloud services that help customers deploy workloads in a flexible, secure, scalable, and highly available way. The AWS infrastructure uses multiple fault isolation constructs to help customers achieve their resilience objectives. These fault isolation boundaries enable customers to design their workloads to take advantage of the predictable scope of impact containment they provide. It's also important to understand how AWS services are designed using these boundaries so that you can make intentional choices about the dependencies you select for your workload.

This paper will first summarize AWS global infrastructure and the fault isolation boundaries it provides, as well as some of the patterns used to design our services. Using this baseline of

understanding, the paper will next outline the different scopes of services AWS provides: zonal, Regional, and global. It will also present best practices for building architectures that use these isolation boundaries and different service scopes to improve the resilience of the workloads you run on AWS. In particular, it provides prescriptive guidance on how to take dependencies on global services while minimizing single points of failure. This will help you make informed choices about your AWS dependencies and how you design your workload for high availability (HA) and disaster recovery (DR).

AWS infrastructure

This section presents a summary of the AWS global infrastructure and the fault isolation boundaries it provides. Additionally, this section will provide an overview of the concept of control planes and data planes, which are critical distinctions in how AWS designs its services. This information provides the baseline to understand how fault isolation boundaries and a service's control plane and data plane apply to the AWS service types we discuss in the next section.

Topics

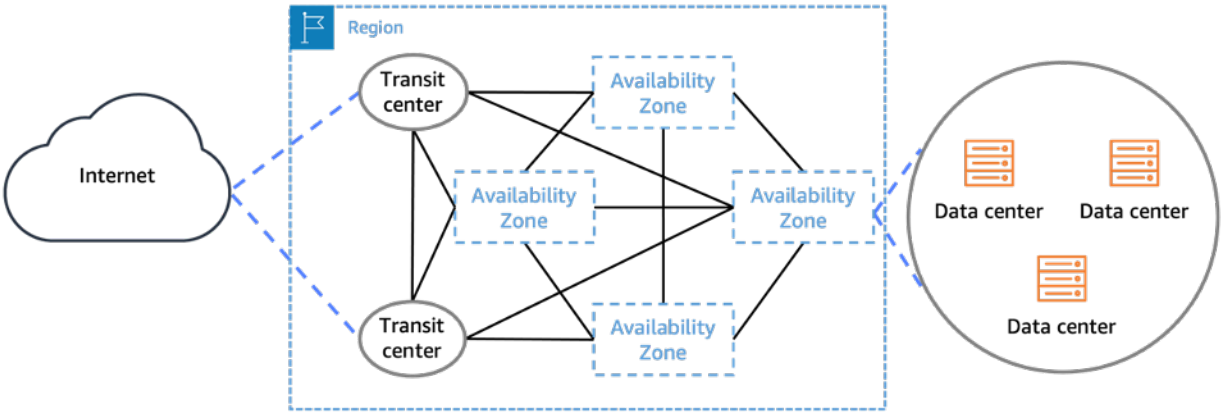
- Availability Zones
- Regions
- AWS Local Zones
- AWS Outposts
- Points of presence
- Partitions
- Control planes and data planes
- Static stability
- Summary

Availability Zones

AWS operates over 100 Availability Zones within several Regions around the world (current numbers can be found here: AWS Global Infrastructure). An Availability Zone is one or more discrete data centers with independent and redundant power infrastructure, networking, and connectivity in an AWS Region. Availability Zones in a Region are meaningfully distant from each other, up to 60 miles (~100 km) to prevent correlated failures, but close enough to use synchronous replication with single-digit millisecond latency. They are designed not to be simultaneously impacted by a shared fate scenario like utility power, water disruption, fiber isolation, earthquakes, fires, tornadoes, or floods. Common points of failure, like generators and cooling equipment, are not shared across Availability Zones and are designed to be supplied by independent power substations. When AWS deploys updates to its services, deployments to Availability Zones in the same Region are separated in time to prevent correlated failure.

All Availability Zones in a Region are interconnected with high-bandwidth, low-latency networking, over fully redundant, dedicated metro fiber. Each Availability Zone in a Region connects to the internet through two transit centers where AWS peers with multiple tier-1 internet providers (for more information, refer to Overview of Amazon Web Services).

These features provide strong isolation of Availability Zones from each other, which we refer to as Availability Zone Independence (AZI). The logical construct of Availability Zones and their connectivity to the internet is depicted in the following figure.



Availability Zones consist of one or more physical data centers that are redundantly connected to each other and the internet

Regions

Each AWS Region consists of multiple independent and physically separate Availability Zones within a geographic area. All Regions currently have three or more Availability Zones. Regions themselves are isolated and independent from other Regions with a few exceptions noted later in this document (refer to Global single-Region operations). This separation between Regions limits service failures, when they occur, to a single Region. Other Regions' normal operations are unaffected in this case. Additionally, the resources and data that you create in one Region do not exist in any other Region unless you explicitly use a replication or copy feature offered by an AWS service or replicate the resource yourself.

Current and planned AWS Regions as of December 2022

AWS Local Zones

AWS Local Zones are a type of infrastructure deployment that places compute, storage, database, and other select AWS services close to large population and industry centers. You can use AWS services, like compute and storage services, in the Local Zone to run low-latency applications at the edge or simplify hybrid cloud migrations. Local Zones have local internet ingress and egress to reduce latency, but are also connected to their parent Region through Amazon's redundant and high-bandwidth private network, giving applications running in AWS Local Zones fast, secure, and seamless access to the full range of services.

AWS Outposts

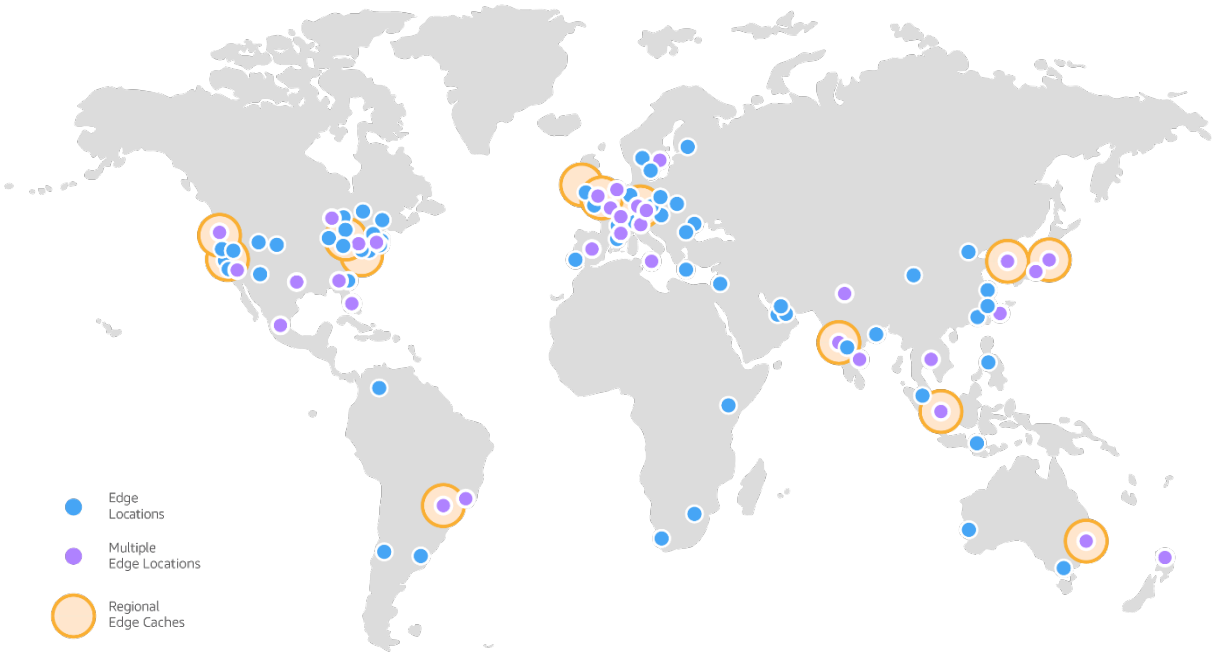
AWS Outposts is a family of fully managed solutions delivering AWS infrastructure and services to virtually any on-premises or edge location for a truly consistent hybrid experience. Outposts solutions allow you to extend and run native AWS services on-premises, and are available in a variety of form factors, from 1U and 2U Outposts servers to 42U Outposts racks, and multiple rack deployments.

With AWS Outposts, you can run select AWS services locally and connect to a broad range of services available in the parent AWS Region. AWS Outposts are fully managed and configurable

compute and storage racks built with AWS-designed hardware that allows customers to run compute and storage on-premises, while seamlessly connecting to AWS's broad array of services in the cloud.

Points of presence

In addition to the AWS Regions and Availability Zones, AWS also operates a globally distributed point of presence (PoP) network. These PoPs host Amazon CloudFront, a content delivery network (CDN); Amazon Route 53, a public Domain Name System (DNS) resolution service; and AWS Global Accelerator (AGA), an edge networking optimization service. The global edge network currently consists of over 410 PoPs, including more than 400 Edge Locations, and 13 regional mid-tier caches in over 90 cities across 48 countries (current status can be found here: Amazon CloudFront Key Features).



Amazon CloudFront global edge network

Each PoP is isolated from the others, which means a failure affecting a single PoP or metropolitan area does not impact the rest of the global network. The AWS network peers with thousands of Tier 1/2/3 telecom carriers globally, is well connected with all major access networks for optimal performance, and has hundreds of terabits of deployed capacity. Edge locations are connected to the AWS Regions through the AWS network backbone, a fully redundant, multiple 100GbE

parallel fiber that circles the globe and links with tens of thousands of networks for improved origin fetches and dynamic content acceleration.

Partitions

AWS groups Regions into partitions. Every Region is in exactly one partition, and each partition has one or more Regions. Partitions have independent instances of AWS Identity and Access Management (IAM) and provide a hard boundary between Regions in different partitions. AWS commercial Regions are in the `aws` partition, Regions in China are in the `aws-cn` partition, and AWS GovCloud Regions are in the `aws-us-gov` partition. Some AWS services are designed to provide cross-Region functionality, such as Amazon S3 Cross-Region Replication or AWS Transit Gateway Inter-Region peering. These types of capabilities are only supported between Regions in the same partition. You cannot use IAM credentials from one partition to interact with resources in a different partition.

Control planes and data planes

AWS separates most services into the concepts of *control plane* and *data plane*. These terms come from the world of networking, specifically routers. The router's data plane, which is its main functionality, is moving packets around based on rules. But the routing policies have to be created and distributed from somewhere, and that's where the control plane comes in.

Control planes provide the administrative APIs used to create, read/describe, update, delete, and list (CRUDL) resources. For example, the following are all control plane actions: launching a new Amazon Elastic Compute Cloud (Amazon EC2) instance, creating an Amazon Simple Storage Service (Amazon S3) bucket, and describing an Amazon Simple Queue Service (Amazon SQS) queue. When you launch an EC2 instance, the control plane has to perform multiple tasks like finding a physical host with capacity, allocating the network interface(s), preparing an Amazon Elastic Block Store (Amazon EBS) volume, generating IAM credentials, adding the Security Group rules, and more. Control planes tend to be complicated orchestration and aggregation systems.

The data plane is what provides the primary function of the service. For example, the following are all parts of the data plane for each of the services involved: the running EC2 instance itself, reading and writing to an EBS volume, getting and putting objects in an S3 bucket, and Route 53 answering DNS queries and performing health checks.

Data planes are intentionally less complicated, with fewer moving parts compared to control planes, which usually implement a complex system of workflows, business logic, and databases.

This makes failure events statistically less likely to occur in the data plane versus the control plane. While both the data and control plane contribute to the overall operation and success of the service, AWS considers them to be distinct components. This separation has both performance and availability benefits.

Static stability

One of the most important resilience characteristics of AWS services is what AWS calls static stability. What this term means is that systems operate in a static state and continue to operate as normal without the need to make changes during the failure or unavailability of dependencies. One way we do this is by preventing circular dependencies in our services that could stop one of those services from successfully recovering. Another way we do this is by maintaining existing state. We consider the fact that control planes are statistically more likely to fail than data planes. Although the data plane typically depends on data that arrives from the control plane, the data plane maintains its existing state and continues working even in the face of control plane impairment. Data plane access to resources, once provisioned, has no dependency on the control plane, and therefore is not affected by any control plane impairment. In other words, even if the ability to create, modify, or delete resources is impaired, existing resources remain available. This makes AWS data planes statically-stable to an impairment in the control plane. You can implement different patterns to be statically-stable against different types of dependency failures.

An example of static stability can be found in Amazon EC2. Once an EC2 instance has been launched, it is just as available as the physical server in a data center. It does not depend on any control plane APIs in order to stay running, or to start running again after a reboot. The same property holds for other AWS resources like VPCs, Amazon S3 buckets and objects, and Amazon EBS volumes.

Static stability is a concept that is deeply ingrained in how AWS designs its services, but it is also a pattern that can be used by customers. In fact, a majority of the best practice guidance for using the different types of AWS services in a resilient way is to implement static stability for production environments. The most reliable recovery and mitigation mechanisms are the ones that require the fewest changes to achieve recovery. Instead of relying on the EC2 control plane to launch new EC2 instances to recover from a failed Availability Zone, having that extra capacity pre-provisioned helps achieve static stability. Thus, eliminating dependencies on control planes (the APIs that implement changes to resources) in your recovery path helps produce more resilient workloads. For more details on static stability, control planes, and data planes, refer to the Amazon Builders' Library article Static stability using Availability Zones.

Summary

AWS utilizes different fault containers in our infrastructure to create fault isolation. The core infrastructure fault containers are partitions, Regions, Availability Zones, control planes, and data planes. Next, we'll examine different types of AWS services, how these fault containers are utilized in their design, and how you should architect workloads with them to be resilient.

AWS service types

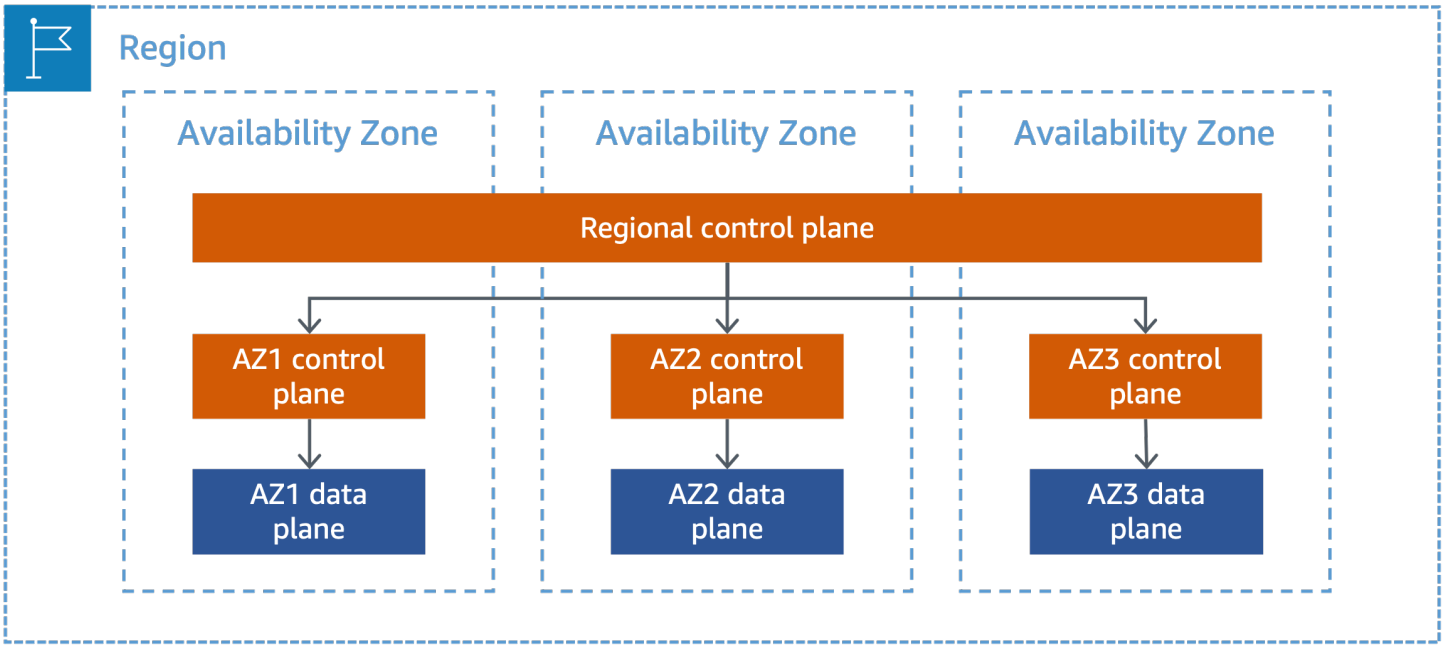
AWS operates three different categories of services based on their fault isolation boundary: zonal, Regional, and global. This section will describe in more detail how these different types of services have been designed so that you can determine how failures within a service of a certain service type will impact your workload running on AWS. It also provides high level guidance on how to architect your workloads to use these services in a resilient way. For global services, this document also provides prescriptive guidance in Appendix A - Partitional service guidance and Appendix B - Edge network global service guidance that can help you prevent impact to your workloads from control plane impairments in AWS services, helping you to safely take dependencies on global services while minimizing introducing single points of failure.

Topics

- Zonal services
- Regional services
- Global services

Zonal services

Availability Zone Independence (AZI) enables AWS to offer zonal services, like Amazon EC2 and Amazon EBS. A zonal service is one that provides the ability to specify which Availability Zone the resources are deployed into. These services operate independently in each Availability Zone within a Region, and more importantly, fail independently in each Availability Zone as well. This means that components of a service in one Availability Zone don't take dependencies on components in other Availability Zones. We can do this because a zonal service has **zonal data planes.** In some cases, such as with EC2, the service also includes zonal control planes for zonally aligned operations, such as launching an EC2 instance. For those services, AWS also provides a regional control plane endpoint to make it easy to interact with the service. The regional control plane also provides Regionally-scoped functionality as well as serves as an aggregation and routing layer on top of the zonal control planes. This is shown in the following figure.



A zonal service with zonally isolated control planes and data planes

Availability Zones give customers the ability to operate production workloads that are more highly available, fault tolerant, and scalable than would be possible from a single data center. When a workload uses multiple Availability Zones, customers are better isolated and protected from issues that impact a single Availability Zone's physical infrastructure. This helps customers to build services that are redundant across Availability Zones and, if architected correctly, remain operational even if one Availability Zone experiences failures. Customers can take advantage of AZI to create highly-available and resilient workloads. Implementing AZI in your architecture helps you to quickly recover from an isolated Availability Zone failure because your resources in one Availability Zones minimize or eliminate interaction with resources in other Availability Zones. This helps remove cross- Availability Zone dependencies which simplifies Availability Zone evacuation. Refer to Advanced Multi-AZ Resilience Patterns for more details on creating Availability Zone evacuation mechanisms. Additionally, you can further take advantage of Availability Zones by following some of the same best practices AWS uses for its own services, such as only deploying changes to a single Availability Zone at a time or removing an Availability Zone from service if a change in that Availability Zone goes badly.

Static stability is also an important concept for Multi-Availability Zone architectures. One of the failure modes you should plan for with Multi-Availability Zone architectures is the loss of an Availability Zone, which can result in the loss of an Availability Zone's worth of capacity. If you haven't pre-provisioned enough capacity to handle the loss of an Availability Zone, this could result in your remaining capacity being overwhelmed by the current load. Additionally, you will need to

depend on the control planes of the zonal services you use to replace that lost capacity, which can be less reliable than a statically-stable design. In this case, pre-provisioning enough extra capacity can help you be statically-stable to the loss of a fault domain, such as an Availability Zone, by being able to continue normal operations without the need for dynamic changes.

You may choose to use an auto scaling group of EC2 instances deployed across multiple Availability Zones to dynamically scale in and out, based on the needs of your workload. Auto scaling works well for gradual changes in usage that occur over minutes to tens of minutes. However, launching new EC2 instances takes time, especially if your instances require bootstrapping (such as installing agents, application binaries, or configuration files). During this time, your remaining capacity could be overwhelmed by the current load. Additionally, deploying new instances through auto scaling relies on the EC2 control plane. This presents a trade-off: To be statically-stable to the loss of a single Availability Zone, you need to pre-provision enough EC2 instances in the other Availability Zones to handle the load that has been shifted away from the impaired Availability Zone, instead of relying on auto scaling to provision new instances. However, pre-provisioning extra capacity can incur additional cost.

For example, during normal operation, let's assume your workload requires six instances to serve customer traffic across three Availability Zones. To be statically-stable against a single Availability Zone failure, you would deploy three instances in each Availability Zone, for a total of nine. If a single Availability Zone-worth of instances failed, you would still have six left and be able to continue to serve your customer traffic without the need to provision and configure new instances during the failure. Achieving static stability for your EC2 capacity has additional cost, since, in this case, you are running 50% additional instances. Not all services where you can pre-provision resources will incur additional cost, such as pre-provisioning an S3 bucket or a user. You will need to weigh any trade-offs of implementing static stability against the risk of exceeding the desired recovery time for your workload.

AWS Local Zones and Outposts bring the data plane of select AWS services closer to end users. The control planes for these services reside in the parent Region. Your Local Zone or Outposts instance will have control plane dependencies for zonal services like EC2 and EBS on the Availability Zone where you created your Local Zone or Outposts subnet. They will also have dependencies on Regional control planes for Regional services like Elastic Load Balancing (ELB), security groups, and the Amazon Elastic Kubernetes Service (Amazon EKS)-managed Kubernetes control plane (if you use EKS). For additional information specific to Outposts, refer to the documentation and support and maintenance FAQ. Implement static stability when using Local Zones or Outposts to help improve resilience to control plane impairments or interruptions in network connectivity to the parent Region.

Regional services

Regional services are services that AWS has built on top of multiple Availability Zones so that customers don't have to figure out how to make the best use of zonal services. We logically group together the service deployed across multiple Availability Zones to present a single Regional endpoint to customers. Amazon SQS and Amazon DynamoDB are examples of Regional services. They use the independence and redundancy of Availability Zones to minimize infrastructure failure as a category of availability and durability risk. Amazon S3, for example, spreads requests and data across multiple Availability Zones and is designed to automatically recover from the failure of an Availability Zone. However, you only interact with the Regional endpoint of the service.

AWS believes that most customers can achieve their resilience goals in a single Region by using Regional services or Multi-AZ architectures that rely on zonal services. However, some workloads may require additional redundancy, and you can use the isolation of AWS Regions to create Multi-Region architectures for HA or business continuity purposes. The physical and logical separation between AWS Regions avoids correlated failures between them. In other words, similar to if you were an EC2 customer and could benefit from the isolation of Availability Zones by deploying across them, you can get that same benefit for Regional services by deploying across multiple Regions. This requires that you implement a multi-Region architecture for your application, which can help you be resilient to the impairment of a Regional service.

However, achieving the benefits of a Multi-Region architecture can be difficult; it requires careful work to take advantage of Regional isolation while not undoing anything at the application level. For example, if you're failing over an application between Regions, you need to maintain strict separation between your application stacks in each Region, be aware of all the application dependencies, and failover all parts of the application together. Achieving this with a complex, microservices-based architecture that has many dependencies between applications requires planning and coordination amongst many engineering and business teams. Allowing individual workloads to make their own failover decisions makes the coordination less complex, but introduces modal behavior through the significant difference in latency that occurs across Regions compared to inside a single Region.

AWS does not provide a synchronous Cross-Region replication feature at this time. When using an asynchronously replicated datastore (provided by AWS) across Regions, there is the possibility of data loss or inconsistency when you fail over your application between Regions. To mitigate possible inconsistencies, you need a reliable data reconciliation process that you have confidence in and may need to operate on multiple data stores across your workload portfolio, or you need to be willing to accept data loss. Finally, you need to practice the failover to know that it will work

when you need it. Regularly rotating your application between Regions to practice failover is a substantial time and resource investment. If you decide to use a synchronously replicated datastore across Regions to support your applications running from more than one Region concurrently, the performance characteristics and latency of such a database that spans 100s or 1000s of miles is very different from a database operating in a single Region. This requires you to plan your application stack from the ground up to account for this behavior. It also makes the availability of both Regions a hard dependency, which could result in decreased resilience of your workload.

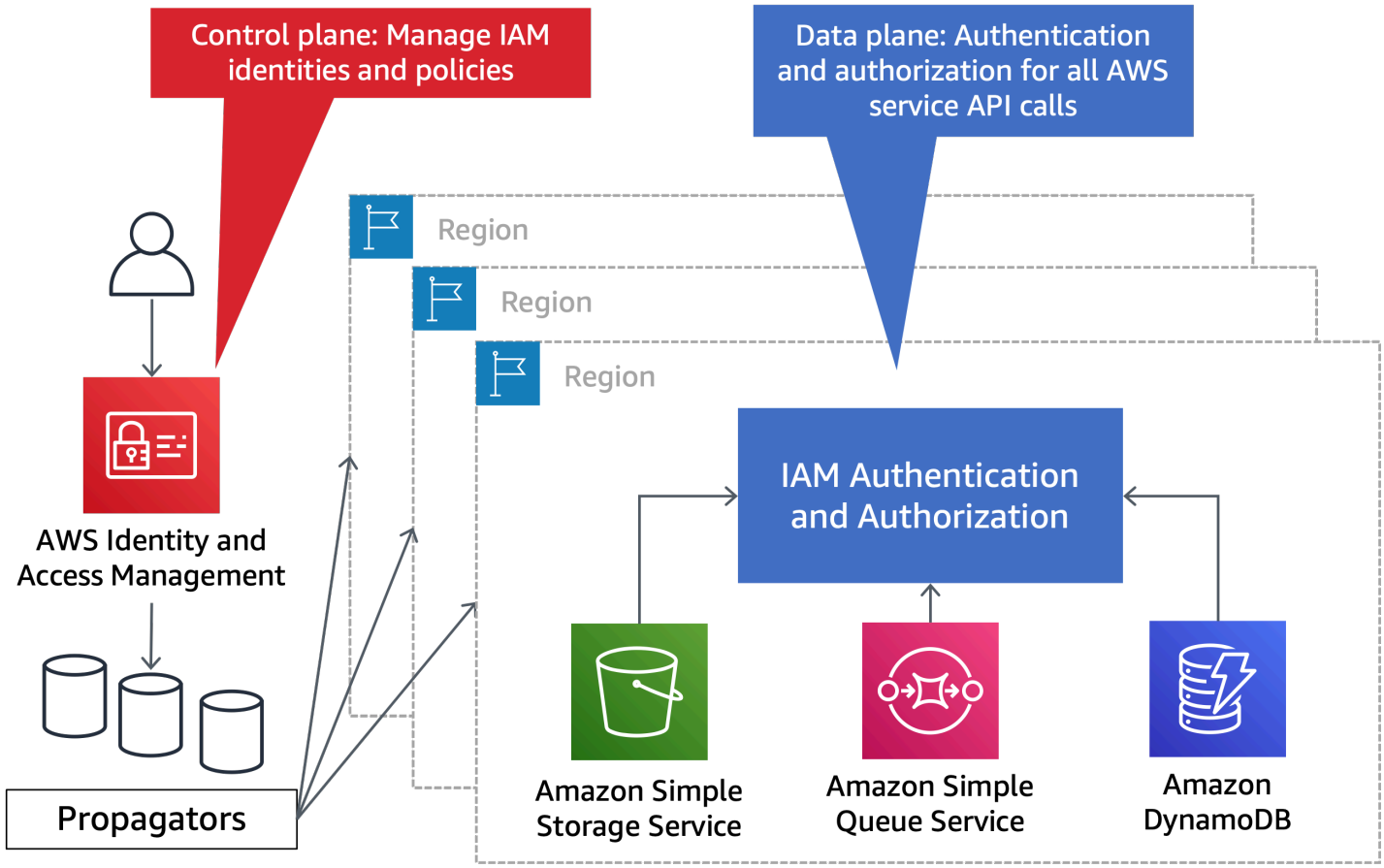
Global services

In addition to Regional and zonal AWS services, there is a small set of AWS services whose control planes and data planes don't exist independently in each Region. Because their resources are not Region-specific, they are commonly referred to as *global*. Global AWS services still follow the conventional AWS design pattern of separating the control plane and data plane in order to achieve static stability. The significant difference for most global services is that their control plane is hosted in a *single* AWS Region, while their data plane is globally distributed. There are three different types of global services and a set of services that can appear to be global based on your selected configuration.

The following sections will identify each type of global service and how their control planes and data planes are separated. You can use this information to guide how you build reliable high availability (HA) and disaster recovery (DR) mechanisms without needing to depend on a global service control plane. This approach helps remove single points of failure in your architecture and avoids potential cross-Region impacts, even when you are operating in a Region that is different from where the global service control plane is hosted. It also helps you safely implement failover mechanisms that do not rely on global service control planes.

Global services that are unique by partition

Some global AWS services exist in each partition (referred to in this paper as *partitional* services). Partitional services provide their control plane in a single AWS Region. Some partitional services, such as AWS Network Manager, are control plane-only and orchestrate the data plane of other services. Other partitional services, such as IAM, have their own data plane that is isolated and distributed across all of the AWS Regions in the partition. Failures in a partitional service do not impact other partitions. In the `aws` partition, the IAM service's control plane is in the `us-east-1` Region, with isolated data planes in each Region of the partition. Partitional services also have independent control planes and data planes in the `aws-us-gov` and `aws-cn` partitions. The separation of control plane and data plane for IAM is shown in the following diagram.



IAM has a single control plane and regionalized data plane

The following are partitional services and their control plane location in the `aws` partition:

- AWS IAM (`us-east-1`)
- AWS Organizations (`us-east-1`)
- AWS Account Management (`us-east-1`)
- Route 53 Application Recovery Controller (ARC) (`us-west-2`) - This service is only present in the `aws` partition
- AWS Network Manager (`us-west-2`)
- Route 53 Private DNS (`us-east-1`)

If any of these service control planes have an availability-impacting event, you may be unable to use the CRUDL-type operations provided by these services. Thus, if your recovery strategy has a dependency on these operations, an availability impact to the control plane or the Region hosting the control plane will reduce your chances of successful recovery. Appendix A - Partitional service

guidance provides strategies for removing dependencies on global service control planes during recovery.

> **Recommendation**
>
> Do not rely on the control planes of partitional services in your recovery path. Instead, rely on the data plane operations of these services. See Appendix A - Partitional service guidance for additional details on how you should design for partitional services.

Global services in the edge network

The next set of global AWS services have a control plane in the `aws` partition and host their data planes in the global points of presence (PoP) infrastructure (and potentially AWS Regions as well). The data planes hosted in PoPs can be accessed from resources in any partition as well as the internet. For example, Route 53 operates its control plane in the `us-east-1` Region, but its data plane is distributed across hundreds of PoPs globally, as well as each AWS Region (to support Route 53 Public and Private DNS within the Region). Route 53 health checks are also part of the data plane, and are performed from eight AWS Regions in the `aws` partition. Clients can resolve DNS using Route 53 public hosted zones from anywhere on the internet, including other partitions like GovCloud, as well as from an AWS Virtual Private Cloud (VPC). The following are global edge network services and their control plane location in the `aws` partition:

- Route 53 Public DNS (`us-east-1`)
- Amazon CloudFront (`us-east-1`)
- AWS WAF Classic for CloudFront (`us-east-1`)
- AWS WAF for CloudFront (`us-east-1`)
- Amazon Certificate Manager (ACM) for CloudFront (`us-east-1`)
- AWS Global Accelerator (AGA) (`us-west-2`)
- AWS Shield Advanced (`us-east-1`)

If you use AGA health checks for EC2 instances or Elastic IP addresses, these use Route 53 health checks. Creating or updating AGA health checks would depend on the Route 53 control plane in `us-east-1`. The execution of the AGA health checks utilizes the Route 53 health check data plane.

During a failure impacting the Region hosting the control planes for these services, or a failure impacting the control plane itself, you may be unable to use the CRUDL-type operations provided by these services. If you have taken dependencies on these operations in your recovery strategy, that strategy may be less likely to succeed than if you only rely on the data plane of these services.

Recommendation

Do not rely on the control plane of edge network services in your recovery path. Instead, rely on the data plane operations of these services. See Appendix B - Edge network global service guidance for additional details on how to design for global services in the edge network.

Global Single-Region operations

The final category is composed of specific control plane operations within a service that have a global impact scope, not entire services like the previous categories. While you interact with zonal and Regional services in the Region you specify, certain operations have an underlying dependency on a single Region that is different from where the resource is located. These are different than services that are only provided in a single Region; refer to Appendix C - Single-Region services for a list of those services.

During a failure impacting the underlying global dependency, you may be unable to use the CRUDL-type actions of the dependent operations. If you have taken dependencies on these operations in your recovery strategy, that strategy may be less likely to succeed than if you only rely on the data plane of these services. You should avoid dependencies on these operations for your recovery strategy.

The following is a list of services that other services may take dependencies on, which have global scope:

- **Route 53**

  Several AWS services create resources that provide a resource-specific DNS name(s). For example, when you provision an Elastic Load Balancer (ELB), the service creates public DNS records and health checks in Route 53 for the ELB. This relies on the Route 53 control plane in `us-east-1`. Other services that you use may also need to provision an ELB, create public Route 53 DNS records, or create Route 53 health checks as part of their control plane workflows. For example, provisioning an Amazon API Gateway REST API resource, an Amazon Relational Database Service

(Amazon RDS) database, or an Amazon OpenSearch Service domain all result in creating DNS records in Route 53. The following is a list of services whose control plane depends on the Route 53 control plane in `us-east-1` to create, update, or delete DNS records, hosted zones, and/or create Route 53 health checks. This list is not exhaustive; it is meant to highlight some of the most commonly-used services whose control plane actions for creating, updating, or deleting resources depend on the Route 53 control plane:

- Amazon API Gateway REST and HTTP APIs
- Amazon RDS instances
- Amazon Aurora databases
- Amazon ELB load balancers
- AWS PrivateLink VPC endpoints
- AWS Lambda URLs
- Amazon ElastiCache
- Amazon OpenSearch Service
- Amazon CloudFront
- Amazon MemoryDB for Redis
- Amazon Neptune
- Amazon DynamoDB Accelerator (DAX)
- AGA
- Amazon Elastic Container Service (Amazon ECS) with DNS-based Service Discovery (which uses the AWS Cloud Map API to manage Route 53 DNS)
- Amazon EKS Kubernetes control plane

It is important to note that the VPC DNS service for EC2 instance hostnames exists independently in each AWS Region and does not depend on the Route 53 control plane. Records that AWS creates for EC2 instances in the VPC DNS service, like `ip-10-0-10.ec2.internal`, `ip-10-0-1-5.compute.us-west-2.compute.internal`, `i-0123456789abcdef.ec2.internal`, and `i-0123456789abcdef.us-west-2.compute.internal`, do not rely on the Route 53 control plane in `us-east-1`.

Recommendation

Do not rely on creating, updating, or deleting resources that require the creation, updating, or deletion of Route 53 resource records, hosted zones, or health checks in

your recovery path. Pre-provision these resources, like ELBs, to prevent a dependency on the Route 53 control plane in your recovery path.

- **Amazon S3**

  The following Amazon S3 control plane operations have an underlying dependency on `us-east-1` in the `aws` partition. A failure impacting Amazon S3 or other services in `us-east-1` could cause these control planes actions to be impaired in other Regions:

```
PutBucketCors
DeleteBucketCors
PutAccelerateConfiguration
PutBucketRequestPayment
PutBucketObjectLockConfiguration
PutBucketTagging
DeleteBucketTagging
PutBucketReplication
DeleteBucketReplication
PutBucketEncryption
DeleteBucketEncryption
PutBucketLifecycle
DeleteBucketLifecycle
PutBucketNotification
PutBucketLogging
DeleteBucketLogging
PutBucketVersioning
PutBucketPolicy
DeleteBucketPolicy
PutBucketOwnershipControls
DeleteBucketOwnershipControls
PutBucketAcl
PutBucketPublicAccessBlock
DeleteBucketPublicAccessBlock
```

  The control plane for Amazon S3 Multi-Region Access Points (MRAP) is hosted only in `us-west-2` and requests to create, update, or delete MRAPs target that Region directly. The control plane for MRAP also has underlying dependencies on AGA in `us-west-2`, Route 53 in `us-east-1`, and ACM in each Region where the MRAP is configured to serve content from. You should not depend on the availability of the MRAP control plane in your recovery path or in your

own systems' data planes. This is distinct from MRAP failover controls that are used to specify active or passive routing status for each of your buckets in the MRAP. These APIs are hosted in five AWS Regions and can be used to effectively shift traffic using the service's data plane.

Additionally, Amazon S3 bucket names are globally unique and all calls to the `CreateBucket` and `DeleteBucket` APIs depend on `us-east-1`, in the `aws` partition, to ensure name uniqueness, even though the API call is directed at the specific Region in which you want to create the bucket. Finally, if you have critical bucket creation workflows, you should not depend on the availability of any specific spelling of a bucket name, particularly those following a discernible pattern.

> **Recommendation**
>
> Do not rely on deleting or creating new S3 buckets or updating S3 bucket configurations as part of your recovery path. Pre-provision all required S3 buckets with the necessary configurations so that you do not need to make changes in order to recover from a failure. This approach applies to MRAPs as well.

- **CloudFront**

Amazon API Gateway provides edge-optimized API endpoints. Creating these endpoints depends on the CloudFront control plane in `us-east-1` to create the distribution in front of the gateway endpoint.

> **Recommendation**
>
> Do not rely on creating new edge-optimized API Gateway endpoints as part of your recovery path. Pre-provision all required API Gateway endpoints.

All of the dependencies discussed in this section are control plane actions, not data plane actions. If your workloads are configured to be statically-stable, these dependencies should not impact your recovery path, keeping in mind that static stability requires additional work or services to implement.

Services that use default global endpoints

In a few cases, AWS services provide a default, global endpoint, like AWS Security Token Service (AWS STS). Other services may use this default, global endpoint in their default configuration. This means that a Regional service you are using could have a global dependency on a single AWS Region. The following details explain how to remove unintended dependencies on default global endpoints that will help you use the service in a Regional way.

AWS STS: STS is a web service that enables you to request temporary, limited-privilege credentials for IAM users or for users you authenticate (federated users). STS usage from the AWS software development kit (SDK) and command line interface (CLI) defaults to `us-east-1`. The STS service also provides Regional endpoints. These endpoints are enabled by default in Regions that are also enabled by default. You can take advantage of these at any time by configuring your SDK or CLI following these directions: AWS STS Regionalized endpoints. Using SigV4A also requires temporary credentials requested from a Regional STS endpoint. You cannot use the global STS endpoint for this operation.

> **Recommendation**
>
> Update your SDK and CLI configuration to use the Regional STS endpoints.

Security Assertion Markup Language (SAML) Sign-in: SAML services exist in all AWS Regions. To use this service, choose the appropriate regional SAML endpoint, like https://us-west-2.signin.aws.amazon.com/saml. You must make updates to configurations in your trust policies and Identity Provider (IdP) to use the regional endpoints. Refer to the AWS SAML documentation for specific details.

If you are using an IdP that is also hosted on AWS, there is a risk that they may also be impacted during an AWS failure event. This could result in you not being able to update your IdP configuration or you may be unable to federate entirely. You should pre-provision “break-glass” users in case your IdP is impaired or unavailable. Refer to Appendix A - Partitional service guidance for details on how to create break-glass users in a statically-stable way.

> **Recommendation**
>
> Update your IAM role trust policies to accept SAML logins from multiple Regions. During a failure, update your IdP configuration to use a different Regional SAML endpoint if your

preferred endpoint is impaired. Create a break-glass user(s) in case your IdP is impaired or unavailable.

AWS IAM Identity Center: Identity Center is a cloud-based service that makes it easy to centrally manage single sign-on access to a customer's AWS accounts and cloud applications. Identity Center must be deployed in a single Region of your choosing. However, the default behavior for the service is to use the global SAML endpoint (https://signin.aws.amazon.com/saml), which is hosted in `us-east-1`. If you have deployed Identity Center into a different AWS Region, you should update the relaystate URL of every permission set to target the same Regional console endpoint as your Identity Center deployment. For example, if you deployed Identity Center into `us-west-2`, you should update the relaystate of your permissions sets to use https://us-west-2.console.aws.amazon.com. This will remove any dependency on `us-east-1` from your Identity Center deployment.

Additionally, because IAM Identity Center can only be deployed into a single Region, you should pre-provision "break-glass" users in case your deployment is impaired. Refer to Appendix A - Partitional service guidance for details on how to create break-glass users in a statically-stable way.

Recommendation

Set the relaystate URL of your permission sets in IAM Identity Center to match the Region where you have the service deployed. Create a break-glass user(s) in case your IAM Identity Center deployment is unavailable.

Amazon S3 Storage Lens: Storage Lens provides a default dashboard called default-account-dashboard. The dashboard configuration and its associated metrics are stored in `us-east-1`. You can create additional dashboards in other Regions by specifying the home Region for the dashboard configuration and metric data.

Recommendation

If you require data from the default S3 Storage Lens dashboard during a failure impacting the service in `us-east-1`, create an additional dashboard in an alternate home Region. You can also duplicate any other custom dashboards you have created in additional Regions.

Global services summary

The data planes for global services apply similar isolation and independence principles as Regional AWS services. A failure impacting the data plane of IAM in a Region doesn't affect the operation of the IAM data plane in another AWS Region. Similarly, a failure impacting the data plane of Route 53 in a PoP doesn't affect the operation of the Route 53 data plane in the rest of the PoPs. Therefore, what we must consider are service availability events that affect the Region where the control plane operates or affect the control plane itself. Because there is only a single control plane for each global service, a failure affecting that control plane could have cross-Region effects on CRUDL-type operations (which are the configuration operations that are typically used to set up or configure a service as opposed to the direct use of the service).

The most effective way to architect workloads to use global services resiliently is to use static stability. During a failure scenario, design your workload not to need to make changes with a control plane to mitigate the impact or failover to a different location. Refer to Appendix A - Partitional service guidance and Appendix B - Edge network global service guidance for prescriptive guidance on how to utilize these types of global services in order to remove control plane dependencies and eliminate single points of failure. If you require the data from a control plane operation for recovery, cache this data in a data store that can be accessed through its data plane, like an AWS Systems Manager Parameter Store (SSM Parameter Store) parameter, a DynamoDB table, or an S3 bucket. For redundancy, you may also choose to store that data in an additional Region. For example, following the best practices for Route 53 Application Recovery Controller (ARC), you should hardcode or bookmark your five Regional cluster endpoints. During a failure event, you might not be able to access some API operations, including Route 53 ARC API operations that are not hosted on the extremely reliable data plane cluster. You can list the endpoints for your Route 53 ARC clusters by using the `DescribeCluster` API operation.

The following is a summary of some of the most common misconfigurations or anti-patterns that introduce dependencies on global services' control planes:

- Making changes to Route 53 records, like updating an A record's value or changing a weighted record set's weights, to perform failover.
- Creating or updating IAM resources, including IAM roles and policies, during a failover. This typically isn't intentional, but might be a result of an untested failover plan.
- Relying on IAM Identity Center for operators to gain access to production environments during a failure event.

- Relying on the default IAM Identity Center configuration to utilize the console in `us-east-1` when you have deployed Identity Center into a different Region.
- Making changes to AGA traffic dial weights to manually perform a Regional failover.
- Updating a CloudFront distribution's origin configuration to fail away from an impaired origin.
- Provisioning disaster recovery (DR) resources, like ELBs and RDS instances during a failure event, that depend on creating DNS records in Route 53.

The following is a summary of the recommendations provided in this section for using global services in a resilient way that would help prevent the previous common anti-patterns.

Recommendation summary

Do not rely on the control planes of partitional services in your recovery path. Instead, rely on the data plane operations of these services. See Appendix A - Partitional service guidance for additional details on how you should design for partitional services.
Do not rely on the control plane of edge network services in your recovery path. Instead, rely on the data plane operations of these services. See Appendix B - Edge network global service guidance for additional details on how to design for global services in the edge network.
Do not rely on creating, updating, or deleting resources that require the creation, updating, or deletion of Route 53 resource records, hosted zones, or health checks in your recovery path. Pre-provision these resources, like ELBs, to prevent a dependency on the Route 53 control plane in your recovery path.
Do not rely on deleting or creating new S3 buckets or updating S3 bucket configurations as part of your recovery path. Pre-provision all required S3 buckets with the necessary configurations so that you do not need to make changes in order to recover from a failure. This approach applies to MRAPs as well.
Do not rely on creating new edge-optimized API Gateway endpoints as part of your recovery path. Pre-provision all required API Gateway endpoints.
Update your SDK and CLI configuration to use the Regional STS endpoints.
Update your IAM role trust policies to accept SAML logins from multiple Regions. During a failure, update your IdP configuration to use a different Regional SAML endpoint if your preferred endpoint is impaired. Create break-glass users in case your IdP is impaired or unavailable.

Set the relaystate URL of your permission sets in IAM Identity Center to match the Region where you have the service deployed. Create a break-glass user(s) in case your Identity Center deployment is unavailable.
If you require data from the default S3 Storage Lens dashboard during a failure impacting the service in `us-east-1`, create an additional dashboard in an alternate home Region. You can also duplicate any other custom dashboards you have created in additional Regions.

Conclusion

AWS provides several different constructs for fault isolation boundaries. You should consider how you architect for zonal, Regional, and global services as well as the potential impacts on your workload and your workload's ability to recover during control plane impairments. Static stability is one of the primary ways that you can avoid control plane dependencies and create reliable and resilient HA and DR mechanisms when you use AWS services.

Appendix A - Partitional service guidance

For partitional services, you should implement static stability in order to maintain resilience of your workload during an AWS service control plane impairment. The following provides prescriptive guidance on how to consider dependencies on partitional services as well as what will and may not work during a control plane impairment.

AWS Identity and Access Management (IAM)

The AWS Identity and Access Management (IAM) control plane consists of all public IAM APIs (including Access Advisor but not Access Analyzer or IAM Roles Anywhere). This includes actions like `CreateRole`, `AttachRolePolicy`, `ChangePassword`, `UpdateSAMLProvider`, and `UpdateLoginProfile`. The IAM data plane provides authentication and authorization for IAM principals in each AWS Region. During a control plane impairment, CRUDL type operations for IAM may not work, but authentication and authorization for existing principals will continue to work. STS is a data plane-only service that is separate from IAM, and does not depend on the IAM control plane.

What this means is that when you are planning for dependencies on IAM, you should not rely on the IAM control plane in your recovery path. For example, a statically-stable design for a “break-glass” admin user would be to create a user with the appropriate permissions attached, have the password set and the access key and secret access key provisioned, and then lock those credentials in a physical or virtual vault. When required during an emergency, retrieve the user credentials from the vault and use them as needed. A non-statically-stable design would be to provision the user during a failure, or having the user pre-provisioned, but only attaching the admin policy when required. These approaches would depend on the IAM control plane.

AWS Organizations

The AWS Organizations control plane consists of all public Organizations APIs like `AcceptHandshake`, `AttachPolicy`, `CreateAccount`, `CreatePolicy`, and `ListAccounts`. There is not a data plane for AWS Organizations. It orchestrates the data plane for other services like IAM. During a control plane impairment, CRUDL type operations for Organizations may not work, but the policies, like Service Control Policies (SCP) and Tag Policies, will continue to work and be evaluated as part of the IAM authorization process. Delegated admin capabilities and multi-

account features in other AWS services that are supported by Organizations will also continue to work.

What this means is that when you are planning for dependencies on AWS Organizations, you should not rely on the Organizations control plane in your recovery path. Instead, implement static stability in your recovery plan. For example, a non-statically-stable approach might be to update SCPs to remove restrictions on allowed AWS Regions via the `aws:RequestedRegion` condition, or to enable admin permissions for specific IAM roles. This relies on the Organizations control plane to make these updates. A better approach would be to use session tags to grant the use of admin permissions. Your Identity Provider (IdP) can include session tags that can be evaluated against the `aws:PrincipalTag` condition, which helps you to dynamically configure permissions for certain principals while helping your SCPs to remain static. This removes dependencies on control planes and only utilizes data plane actions.

AWS Account Management

The AWS Account Management control plane is hosted in us-east-1 and consists of all public APIs for managing an AWS account, such as `GetContactInformation` and `PutContactInformation`. It also includes creating or closing a new AWS account through the management console. The APIs for `CloseAccount`, `CreateAccount`, `CreateGovCloudAccount`, and `DescribeAccount` are part of the AWS Organizations control plane, which is also hosted in us-east-1. Additionally, creating a GovCloud account outside of AWS Organizations relies on the AWS account management control plane in us-east-1. Also, GovCloud accounts must be 1:1 linked to an AWS account in the `aws` partition. Creating accounts in the `aws-cn` partition does not rely on us-east-1. The data plane for AWS accounts is the accounts themselves. During a control plane impairment, CRUDL-type operations (like creating a new account or getting and updating contact information) for AWS accounts may not work. References to the account in IAM policies will continue to work.

What this means is that when you are planning for dependencies on AWS Account Management, you should not rely on the Account Management control plane in your recovery path. Although the Account Management control plane doesn't provide direct functionality that you would typically use in a recovery situation, there may be times when you would. For example, a statically-stable design would be to pre-provision all of the AWS accounts you need for failover. A non-statically-stable design would be to create new AWS accounts during a failure event to host your DR resources.

Route 53 Application Recovery Controller

The control plane for Route 53 ARC consists of the APIs for recovery control and recovery readiness, as identified at: Amazon Route 53 Application Recovery Controller endpoints and quotas. You manage readiness checks, routing controls, and cluster operations by using the control plane. The data plane of ARC is your recovery cluster, which manages the routing control values that are queried by Route 53 health checks, and also implements the safety rules. The data plane functionality of Route 53 ARC is accessed through your recovery cluster APIs like `https://aaaaaaaa.route53-recovery-cluster.eu-west-1.amazonaws.com`.

What this means is that you shouldn't rely on the Route 53 ARC control plane in your recovery path. There are two best practices that help implement this guidance:

- First, bookmark or hard code the five Regional cluster endpoints. This removes the need to use the DescribeCluster control plane operation during a failover scenario to discover the endpoint values.
- Second, use the Route 53 ARC cluster APIs by using the CLI or SDK to perform updates to routing controls and not the AWS Management Console. This removes the management console as a dependency for your failover plan and ensures it depends on only data plane actions.

AWS Network Manager

The AWS Network Manager service is primarily a control plane-only system hosted in us-west-2. Its purpose is to centrally manage the configuration of your AWS Cloud wide area networking (WAN) core network and your AWS Transit Gateway network across AWS accounts, Regions, and on-premises locations. It also aggregates your Cloud WAN metrics in us-west-2, which can also be accessed through the CloudWatch data plane. If Network Manager is impaired, the data plane of the services it orchestrates will not be impacted. The CloudWatch metrics for Cloud WAN are also available in us-west-2. If you want historical metric data, like bytes in and out per Region, to understand how much traffic might shift to other Regions during a failure impacting us-west-2, or for other operational purposes, you can export those metrics as CSV data directly from the CloudWatch console or using this method: Publish Amazon CloudWatch metrics to a CSV file. The data can be found under the `AWS/Network Manager` namespace and you can perform this on a schedule you choose and store it in S3 or in another data store you select. To implement a statically-stable recovery plan, do not use AWS Network Manager to make updates to your network, or rely on data from its control plane operations for failover input.

Route 53 Private DNS

Route 53 private hosted zones are supported in each partition; however, the considerations for private hosted zones and public hosted zones in Route 53 are the same. Refer to *Amazon Route 53* in *Appendix B - Edge network global service guidance*.

Appendix B - Edge network global service guidance

For edge network global services, you should implement static stability in order to maintain resilience of your workload during an AWS service control plane impairment.

Route 53

The Route 53 control plane consists of all public Route 53 APIs covering functionality for hosted zones, records, health checks, DNS query logs, reusable delegation sets, traffic policies, and cost allocation tags. It is hosted in us-east-1. The data plane is the authoritative DNS service, which runs across over 200 PoP locations as well as in each AWS Region, answering DNS queries based on your hosted zones and health check data. Additionally, Route 53 has a data plane for health checks which is also a globally-distributed service across multiple AWS Regions. This data plane performs health checks, aggregates the results, and delivers them to the data planes of Route 53 public and private DNS and AGA. During a control plane impairment, CRUDL-type operations for Route 53 may not work, but DNS resolution and health checks, and updates to routing resulting from changes in health checks, will continue to work.

What this means is that when you are planning for dependencies on Route 53, you should not rely on the Route 53 control plane in your recovery path. For example, a statically-stable design would be to use the status of health checks to perform failovers between Regions or to evacuate an Availability Zone. You can use Route 53 Application Recovery Controller (ARC) routing controls to manually change the status of health checks and alter the responses to DNS queries. There are similar patterns to what ARC provides that you can implement based on your requirements. Some of these patterns are outlined in Creating Disaster Recovery Mechanisms using Route 53 and in the Advanced Multi-AZ Resilience Patterns health check circuit breaker section. If you have elected to use a Multi-Region DR plan, pre-provision resources that require DNS records to be created, like ELBs and RDS instances. A non-statically-stable design would be to update the value of a Route 53 resource record via the `ChangeResourceRecordSets` API, change the weight of a weighted record, or create new records to perform failover. These approaches depend on the Route 53 control plane.

Amazon CloudFront

The Amazon CloudFront control plane consists of all public CloudFront APIs for managing distributions, and is hosted in us-east-1. The data plane is the distribution itself served from the

PoPs in the edge network. It performs the request handling, routing, and caching of your origin content. During a control plane impairment, CRUDL-type operations for CloudFront (including invalidation requests) may not work, but your content will continue to be cached and served, and origin failovers will continue to work.

What this means is that when you are planning for dependencies on CloudFront, you should not rely on the CloudFront control plane in your recovery path. For example, a statically-stable design would be to use automated origin failovers to mitigate the impact from an impairment to one of your origins. You might also choose to build origin load balancing or failover using Lamda@Edge, refer to Three advanced design patterns for high available applications using Amazon CloudFront and Using Amazon CloudFront and Amazon S3 to build multi-Region active-active geo proximity applications for more details on that pattern. A non-statically-stable design would be to manually update the configuration of your distribution in response to an origin failure. This approach would depend on the CloudFront control plane.

Amazon Certificate Manager

If you are using custom certificates with your CloudFront distribution, you also have a dependency on ACM. Using custom certificates with your CloudFront distribution relies on the ACM control plane in the us-east-1 Region. During a control plane impairment, your existing certificates configured in your distribution will continue to work as well as automatic certificate renewals. Do not rely on changing the distribution's configuration or creating new certificates as part of your recovery path.

AWS Web Application Firewall (WAF) and WAF Classic

If you are using AWS WAF with your CloudFront distribution, you have a dependency on the WAF control plane, which is also hosted in the us-east-1 Region. During a control plane impairment, the configured web access control lists (ACLs) and their associated rules continue to function. Do not rely on updating your WAF web ACLs as part of your recovery path.

AWS Global Accelerator

The AGA control plane consists of all public AGA APIs and is hosted in us-west-2. The data plane is the network routing of the anycast IP addresses provided by AGA to your registered endpoints. AGA also utilizes Route 53 health checks to determine the health of your AGA endpoints, which is part of the Route 53 data plane. During a control plane impairment, CRUDL-type operations for AGA may not work. Routing to your existing endpoints, as well as existing health checks, traffic dials,

and endpoint weight configurations used to route or shift traffic to other endpoints and endpoint groups, will continue to work.

What this means is that when you are planning for dependencies on AGA, you should not rely on the AGA control plane in your recovery path. For example, a statically-stable design would be to use the status of the configured health checks to fail away from unhealthy endpoints. Refer to Deploying multi-region applications in AWS using AWS Global Accelerator for examples of this configuration. A non-statically-stable design would be to modify the AGA traffic dial percentages, edit endpoint groups, or remove an endpoint from an endpoint group during an impairment. These approaches would depend on the AGA control plane.

Amazon Shield Advanced

The Amazon Shield Advanced control plane consists of all public Shield Advanced APIs, and is hosted in us-east-1. This includes functionality like `CreateProtection`, `CreateProtectionGroup`, `AssociateHealthCheck`, `DesribeDRTAccess`, and `ListProtections`. The data plane is the DDoS protection provided by Shield Advanced as well as the creation of Shield Advanced metrics. Shield Advanced also utilizes Route 53 health checks (which are part of the Route 53 data plane), if you have configured them. During a control plane impairment, CRUDL-type operations for Shield Advanced may not work, but the DDoS protection configured for your resources, as well as responses to changes in health checks, will continue to function.

What this means is that you should not rely on the Shield Advanced control plane in your recovery path. Although the Shield Advanced control plane doesn't provide direct functionality that you would typically use in a recovery situation, there may be times when you would. For example, a statically-stable design would be to have your DR resources already configured to be part of a protection group and have health checks associated with them as opposed to configuring that protection after the failure occurs. This prevents depending on the Shield Advanced control plane for recovery.

Appendix C - Single-Region services

The following is a list of services, or specific features in that service (which are listed in parentheses after the service name), that are only available in a single Region. The same guidance for implementing static stability provided for other global services applies to these services when you need to plan for dependencies on their control planes and data planes.

- Alexa for Business
- AWS Marketplace (AWS Marketplace Catalog API, AWS Marketplace Commerce Analytics, AWS Marketplace Entitlement Service)
- Billing and Cost Management (AWS Cost Explorer, AWS Cost and Usage Reports, AWS Budgets, Savings Plans)
- AWS BugBust
- Amazon Mechanical Turk
- Amazon Chime
- Amazon Chime SDK (PSTN audio, messaging, identity)
- AWS Chatbot
- AWS DeepRacer
- AWS Device Farm
- Amazon GameSparks
- Amazon Honeycode

Contributors

Contributors to this document include:

- Michael Haken, Principal Solutions Architect, Amazon Web Services

Document revisions

To be notified about updates to this whitepaper, subscribe to the RSS feed.

Change	Description	Date
Minor revision	Updated guide to align with the IAM best practices . For more information, see Security best practices in IAM.	February 9, 2023
Initial publication	Whitepaper published.	November 16, 2022

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.